Cooperative Initialization based Deep Neural Network Training

Pravendra Singh Munender Varshney Vinay P. Namboodiri
Department of Computer Science and Engineering, IIT Kanpur, India
{psingh, munender, vinaypn}@iitk.ac.in

Abstract

Researchers have proposed various activation functions. These activation functions help the deep network to learn non-linear behavior with a significant effect on training dynamics and task performance. The performance of these activations also depends on the initial state of the weight parameters, i.e., different initial state leads to a difference in the performance of a network. In this paper, we have proposed a cooperative initialization for training the deep network using ReLU activation function to improve the network performance. Our approach uses multiple activation functions in the initial few epochs for the update of all sets of weight parameters while training the network. These activation functions cooperate to overcome their drawbacks in the update of weight parameters, which in effect learn better "feature representation" and boost the network performance later. Cooperative initialization based training also helps in reducing the overfitting problem and does not increase the number of parameters, inference (test) time in the final model while improving the performance. Experiments show that our approach outperforms various baselines and, at the same time, performs well over various tasks such as classification and detection. The Top-1 classification accuracy of the model trained using our approach improves by 2.8% for VGG-16 and 2.1% for ResNet-56 on CIFAR-100 dataset.

1. Introduction

Deep neural networks (DNNs) are state-of-the-art models, responsible for transforming research in the area of vision, language and speech [14, 7, 4]. Various works [30, 31, 26, 29, 27, 28, 25, 20, 32] have been proposed for efficient deep learning. These deep network at core performs a linear transformation followed by a non-linear operation using an activation function. The activation function is the one, which is responsible for nonlinear behaviour and the learning capabilities of the network. These activations are non-linear continuous functions which may also possess non-differentiability [22, 18, 8]. Researchers have proposed many activation functions which can be classified into saturated [2, 21, 10] and non-saturated activation functions [2, 18, 35, 22].

The saturated activation belongs to a category, in which the learning process gets slow down due to the very small gradient near-saturated output. These activation functions are experimentally proved to be less effective for training a deep network. The key reason for the failure is the (vanishing/exploding) gradient problems, which mostly occurs due to saturated output in an activation function. This problem is efficiently tackled by using non-saturated activation function, like ReLU [22, 18]. In particular, the derivative of ReLU is one for the positive inputs; hence, the gradient cannot vanish. In contrast, all the negative values are mapped to zero, which restricts the flow of information in DNNs for these negative values. ReLU gets saturated exactly at zero, which makes ReLU fragile at the time of training, and the neuron can die forever. For example, the flow of large gradients through ReLU may update weight parameters in a way that may deactivate neurons for all data points. This problem is known as dying ReLU, which implies that the gradients flow through the neuron will forever be zero from that point. Due to this, the gradient-based optimization algorithm will not be able to update the weights of that neuron unit. Also, training the network on a high learning rate may shoot the number of "dead" neurons in the network as much as 40% of the network [18] (i.e., neurons that never activate across the entire training dataset). So there is a need to set the learning rate properly to reduce the issues.

To resolve these potential problems originated by the hard zero mappings in the ReLU units, various generalizations of ReLU such as Leaky ReLU [18], and PReLU [8] have been proposed. Both Leaky ReLU and PReLU are same as ReLU except for the case of negative inputs in which a small constant slope for Leaky ReLU and a learnable slope for PReLU are used. Similarly, Exponential Linear Units (ELU) [3] is also proposed, which resolves the bias shift [11] from the succeeding layers. ELU [3] gives an exponential value corresponding to negative inputs, which force the mean output of the activation function to reach towards zero. Although ELU is not backed by concrete the-

ory, ELU shows competitive results. Besides all these, Softplus [36] is approximately similar to ReLU, except at zero, where softplus is differentiable and smooth. Softplus is also differentiable everywhere and saturates less, which gives an edge over ReLU. In practice, there is no non-linear activation function that outperforms all the time over all models, datasets, and problems.

In this paper, we propose an approach in which multiple non-linear activation functions are exploited using a cooperative strategy to overcome their drawbacks. We have used multiple activation functions in the initial epochs of training the deep network. The aggregation of gradient from all the activation function gives a *regularization effect* for the gradient flow corresponding to the whole range of inputs (negative values also). This results in regularizing the update of weight parameters, which is a very crucial step in the initial stage. In the next stage, we train the network with only one activation function (standard network) corresponding to each layer of the network. The proposed approach has experimented extensively on different architectures such as ResNet, and VGG-16 models over CIFAR-10, CIFAR-100, and ImageNet datasets. We also experimented with object detection task using SSD-300 on the PASCAL VOC dataset.

Major contributions of this paper are as follows:

- We have shown experimentally that using multiple activation functions in the initial few epochs of the training process benefits the update of the full set of weight parameters, which results in substantial performance improvement later.
- We have shown empirically that a mixture of non-linear activation functions results in significant improvement in the performance as compared to the individual non-linear activation function.
- We have shown that Cooperative Initialization based training also help in reducing overfitting problem.
- Our proposed approach does not increase the number of parameters and inference (test) time in the final model while improving the performance.

2. Previous Work

The first activation function is a step function originally used in the perceptron model [21]. Researchers in the same direction have also proposed many other saturated activation functions such as sigmoid, softmax, and tanh [2]. The ReLU activation further replaces these activation functions, owing to the outstanding performance on deep neural networks [22, 33, 13]. ReLU had escalated the convergence and resolved the vanish gradient problem, normally occurred in saturated activation functions. These activation functions have accelerated the efforts of the research community in solving various vision problems. Several attempts have been made to develop a more efficient network by developing better activation functions, which can resolve the problems arising in the above activation functions. Some variants of ReLU have been proposed such as RReLU, PReLU, leaky ReLU, and others [35, 8, 18].

To resolve the issues of mapping, all negative input to zero (dying ReLU) in the ReLU activation function, Leaky ReLU [18] has been proposed. This mapping causes an information loss (dead neuron), which is resolved by defining a linear function corresponding to negative input, having a small predefined constant slope, to leak some information [18]. However, Leaky ReLU does not give any notable results on performance experimentally. Further, a parametric rectified linear unit (PReLU) has been proposed [8] which uses a learnable slope parameter instead of a constant slope as in Leaky ReLU [18] for negative inputs. PReLU gives better performance than ReLU in many cases. On the second thought, the slope parameter can be randomly sampled from a uniform distribution as used in Randomized Leaky Rectified Linear Unit (RReLU) [35] which reduces the risk of overfitting in the training phase. ELU [3] is also same as ReLU for positive input while it behaves similarly to saturated exponential function for negative inputs. Further, Parametric ELU (PELU) is a scaled version of ELU having a learnable scaling parameter [34]. Most of the above discussed ReLU variants are based on the experiments over the negative inputs of ReLU.

There are few other works in which new activation functions are proposed, such as Maxout (maximum over K affine functions), Softplus, and Adaptive Piecewise Linear (APL) [6, 36, 1]. The APL consist of many non-differentiable points that scale linearly with the number of hinge functions. This will increase the model complexity and affect the parameter updates during back-propagation [14]. In the same manner, Maxout take maximum over multiple feature maps. Softplus is the smooth approximation of ReLU, which is differentiable everywhere.

In this work, we have focused on leveraging benefits from the multiple non-linear activation functions simultaneously. To the best of our knowledge, this is the first work that considers a mixture of non-linear activation functions in the initial few epochs. We have also presented an ablation study and feature visualization to support the proposed approach.

3. Cooperative Network Design (Phase-1)

Nowadays, deep networks consist of huge depth due to the presence of multiple convolutional layers. These convolutional layers are followed by an activation function, which operates on the feature maps (output) of these layers. The training process will update weight parameters using back-propagation. This update solely depends on the be-

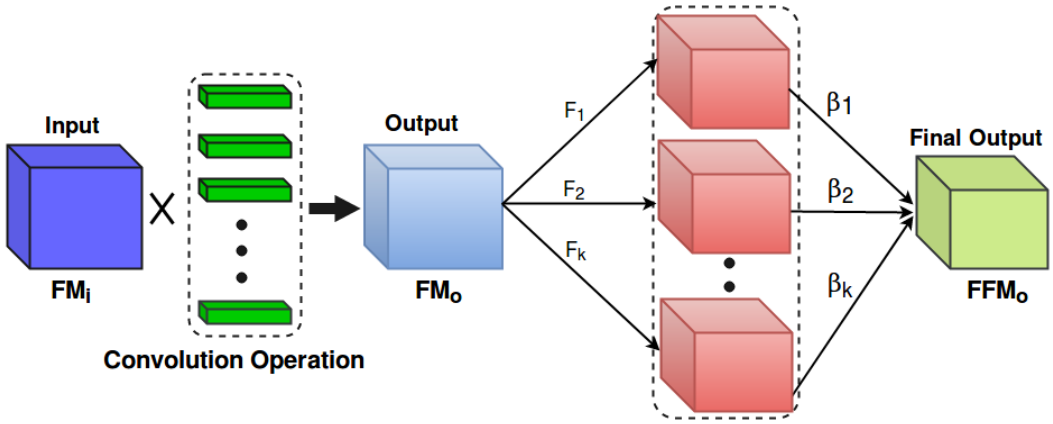

Figure 1. Cooperative design for a particular layer in deep network during Phase-1 training time (Best viewed in color).

havior of the activation function, i.e., ReLU never updates the weight parameter corresponding to negative inputs. Restricting these weights from an update in the initial phase of training may hurt the performance of the deep network later.

To overcome these issues in the network, researchers have proposed various activation functions, which have their advantages and disadvantages. In this work, we have proposed a *cooperative design* having multiple activation functions, helping each other in the initial update of parameters in such a way that all set of weight parameters get an opportunity to contribute to the performance. These activation functions cooperate to overcome their drawbacks and improve the update process for all sets of weight parameters in the initial epochs of training.

In Figure 1, we have shown a block diagram of a layer, where the convolution operation is same as of standard CNN, but we have used k activation functions instead of one activation function. The input feature maps for the given layer are represented as FM_i in Figure 1, which are then convolved using convolutional filters shown in green color. The output feature maps FM_o generated by convolution operation are passed to each activation function, which operates over each element of the given feature maps. This results in the generation of k different sets of feature maps corresponding to each activation function, as shown in Figure 1. The Final output Feature Maps (FFM_o) are the result of the weighted average of these k feature maps as given by the following equation:

$$FFM_o = \beta_1 F_1(FM_o) + \beta_2 F_2(FM_o) + ... + \beta_k F_k(FM_o);$$

where $F_1, F_2, ...F_k$ are the activation functions applied on the FM_o feature map. The final output feature maps (FFM_o) are the weighted average of outputs from these activation function when applied to input feature maps (shown in red color). The corresponding weights for averaging are $\beta_1, \beta_2, ..., \beta_k$ where $1, 2, ..., k$ represents the k activation functions. We assume that each activation contributes equally in the update process (improvement) of the weight parameters by assigning equal weight to each parameter ($\beta_i = 1/k$).

We have presented an ablation study to shown empirically that even if we train the model completely by using a mixture of activation functions, it will results in substantial performance improvement as compare to the individual activation function.

4. Standard Network Design (Phase-2)

In Phase-2, the network use only a single activation function, i.e., ReLU activation for each layer in the model, instead of the mixture of activation functions. In the training process of phase-1 design, our focus is to shift the model in a stable state where all sets of model parameters are in a better state than a randomly initialized state. The phase-1 network design is trained for a few epochs such that the mixture of activation functions will overcome the drawbacks of each other by cooperation in the update process of weight parameters. All the k activation functions will get different gradients and averaging them give a *regularization effect*, which updates all sets of parameters uniformly without undermining others.

In Phase-2, We use only one activation function, so we opt to use only ReLU as our activation function. ReLU is a linear (identity) mapping for all positive values and zero values for all negative values. ReLU activation shows a sparse behavior as all negative inputs are mapped to zero. After getting into a better state from phase-1, we often desire to make feature maps sparse enough, which is easily possible with ReLU as it possesses many deactivated neurons, giving a regularized effect to the model. Sparsity results in concise models that often have better predictive power and fewer chances of overfitting/noise. In a sparse network, all neurons cannot be activated simultaneously in a model. Only those set of neurons get activated which are responsible for a particular aspect of the given task, e.g.,

there are a given set of neurons which gets activated for a face like structure in a human detecting task while the same set of neurons are inactive for other parts of the body. That's the reason standard ReLU seems to be less prone to overfitting vs. leaky ReLU with modern architectures. We have also presented an ablation study to empirically show that ReLU in Phase-2 will results in substantially improved performance as compare to other possible options. Hence ReLU is competing with all other non-linear activation functions in Phase-2.

5. Training Method

The proposed training framework is divided into two phases corresponding to the two network design phases. For Phase-1 (Cooperative) training, we use 20% of epochs used in standard (Phase-2) training with a mixture of non-linear activation functions (having equal weight). We have presented an ablation on the number of epochs used in Phase-1 training to validate the choice mentioned above.

In the phase-1 training process, gradient calculation in the back-propagation algorithm [15] is the aggregate of the gradient calculated for each activation function. This gives a regularized effect on the gradient and provides equal opportunity for all weight parameters to optimize. Phase-2 training is the same as the standard training of the model with only one non-linear activation function (ReLU). Please note that a mixture of activation functions is used in only Phase-1 training, while Phase-2 has only ReLU at every layer. Further details are provided in the experimental section.

6. Experiments and Results

In this section, we have evaluated the performance of the proposed approach on classification and detection task. Our experimentation uses the state-of-art CNN architectures such as ResNet [9], and VGG-16 [24] for various activation functions. All these models are trained over three standard benchmark datasets: CIFAR10, CIFAR100 [12] and ImageNet [13] dataset. We have also performed experiments using SSD [17] on PASCAL VOC [5] for object detection.

In these experiments, we have used four most prominently used non-linear activation functions (ReLU, PReLU, ELU, and SoftPlus). We have preferred PReLU to resolve the issue of dying ReLU; however, someone may also prefer Leaky ReLU. ELU has an exponential function for negative input, which is contrary to ReLU. This behavior of ELU pushes the mean to the neighborhood of zero, similar to the case of batch normalization [11]. This shift of mean toward the vicinity of zero accelerate the training of network (fast convergence). ELU also guarantees more robustness towards the noise. These are the few reasons we have selected ELU in the mixture of activation functions. The behavior of ReLU and Softplus [36] is almost similar, excluding near the periphery of zero, where the softplus is differentiable and smooth. Softplus has privilege over ReLU due to differentiability in the entire domain, and it saturates less.

The scope of activation functions depends on the problems. We have selected ReLU, PReLU, ELU, and SoftPlus, which are the most widely used non-linear activation functions in image classification and object detection problems.

In our experiments, baselines (using activation function ReLU/PReLU/ELU/SoftPlus) have been reproduced using a standard training procedure in PyTorch [23] framework. We trained these models using 300 and 90 epochs for CIFAR and ImageNet dataset respectively.

6.1. CIFAR 10 and CIFAR 100

The CIFAR-10 and CIFAR-100 [12] are the datasets having tiny natural images. CIFAR10 datasets have 10 different image classes, while CIFAR-100 datasets have 100 classes. There are 50,000 training images and 10,000 test images, where all images are RGB images with a dimension of 32x32.

In the experiments with the CIFAR dataset, we perform standard data augmentation methods of random cropping to a size of 32×32 and random horizontal flipping. The optimization is performed using Stochastic Gradient Descent (SGD) algorithm with momentum 0.9 and a minibatch size of 64. In Phase-2 training, the initial learning rate is set to 0.1, which is decreased by a factor of 5 after every 50 epoch. The models are trained from scratch for around 300 epochs. For Phase-1 (Cooperative) training, we use only 20% of epochs used in Phase-2 training with PReLU, ReLU, ELU, and SoftPlus activation functions (having equal weight). The learning rate is set to 0.1 and is decreased by a factor of 5 after every 10 epoch in Phase-1 (Cooperative) training. For evaluation, the validation images are used. The results on the CIFAR-10/100 datasets for all the architectures have been reproduced in the PyTorch framework.

The results are shown in Table 1, 2. We observe a consistent improvement in accuracy for VGG-16 and ResNet-56 over CIFAR. The model trained with our proposed two-phase training procedure not only outperforms ReLU significantly but also other non-linear activation functions such as PReLU, ELU, and SoftPlus, as shown in Table 1, 2.

6.2. ImageNet

ImageNet dataset [13] contains 1000 classes, each category roughly having 1000 images. The dataset contains about 1.2 million training images, 50,000 validation images, and 100,000 test images (with no labels). The training is performed on training data, whereas all evaluations are performed on the validation set.

Model	Activation Function	Accuracy(%)
VGG-16 (Baseline)	ReLU	93.6
VGG-16 (Baseline)	PReLU	93.7
VGG-16 (Baseline)	SoftPlus	90.5
VGG-16 (Baseline)	ELU	92.3
VGG-16 (Ours)	**Mix (ReLU)**	**94.2**
ResNet-56 (Baseline)	ReLU	93.5
ResNet-56 (Baseline)	PReLU	94.0
ResNet-56 (Baseline)	SoftPlus	92.0
ResNet-56 (Baseline)	ELU	92.0
ResNet-56 (Ours)	**Mix (ReLU)**	**94.4**

Table 1. Classification accuracies for CIFAR 10. All baselines have been reproduced using corresponding activation function in PyTorch framework. Activation Function: Mix (ReLU) means model is trained using a mixture of Activation Functions in Phase-1 (Cooperative) training and then in Phase-2 model is trained using ReLU Activation Function.

Model	Activation Function	Accuracy(%)
VGG-16 (Baseline)	ReLU	72.0
VGG-16 (Baseline)	PReLU	72.5
VGG-16 (Baseline)	SoftPlus	64.2
VGG-16 (Baseline)	ELU	66.9
VGG-16 (Ours)	**Mix (ReLU)**	**74.0**
ResNet-56 (Baseline)	ReLU	71.6
ResNet-56 (Baseline)	PReLU	71.9
ResNet-56 (Baseline)	SoftPlus	69.3
ResNet-56 (Baseline)	ELU	69.5
ResNet-56 (Ours)	**Mix (ReLU)**	**73.1**

Table 2. Classification accuracies for CIFAR 100. All baselines have been reproduced using corresponding activation function in PyTorch framework. Activation Function: Mix (ReLU) means model is trained using a mixture of Activation Functions in Phase-1 (Cooperative) training and then in Phase-2 model is trained using ReLU Activation Function.

Model	Activation Function	Accuracy(%)
AlexNet (Baseline)	ReLU	56.6
AlexNet (Baseline)	PReLU	56.9
AlexNet (Baseline)	SoftPlus	55.2
AlexNet (Baseline)	ELU	56.6
AlexNet (Ours)	**Mix (ReLU)**	**57.2**
ResNet-18 (Baseline)	ReLU	69.8
ResNet-18 (Baseline)	PReLU	69.1
ResNet-18 (Baseline)	SoftPlus	68.8
ResNet-18 (Baseline)	ELU	68.2
ResNet-18 (Ours)	**Mix (ReLU)**	**70.8**

Table 3. Classification accuracies for ImageNet. The accuracy is reported over validation set using 1-crop setting (https://pytorch.org/docs/stable/torchvision/models.html).

For ImageNet experiments, we perform standard data augmentation methods of random cropping to a size of 224×224 and random horizontal flipping. For optimization, Stochastic Gradient Descent (SGD) is used with momentum

Class	**SSD (Baseline) AP**	**SSD Mix (ReLU) AP**
aero	80.40	82.53
bike	82.95	82.54
bird	74.62	77.02
boat	71.61	72.45
bottle	50.49	51.36
bus	86.04	85.57
car	86.55	86.28
cat	88.02	86.91
chair	60.88	63.23
cow	83.10	81.58
table	77.87	78.35
dog	85.55	84.06
horse	86.68	87.79
mbike	84.14	85.85
person	78.26	79.29
plant	50.44	52.82
sheep	74.28	78.13
sofa	80.03	80.72
train	85.88	87.28
tv	75.49	77.15
mAP	77.16	**78.05**

Table 4. AP for each class with Baseline SSD-300 (using standard training schedule) and SSD-300 Mix (ReLU) using our proposed training schedule on VOC2007 test dataset. Training data, 07+12 is the union of the VOC2007 and VOC2012 trainval dataset.

0.9 and a minibatch size of 256. For Phase-2 training, the initial learning rate is set to 0.1, which is decreased by a factor of 10 after every 30 epoch. The models are trained for around 90 epochs. The evaluation is done over validation images are subjected to center cropping of size 224×224. For Phase-1 (Cooperative) training, we use 20% of epochs used in Phase-2 training with PReLU, ReLU, ELU, and SoftPlus activation functions (having equal weight). The learning rate is set to 0.1 and is decreased by a factor of 10 after every 5 epoch in Phase-1 (Cooperative) training.

The results are shown in Table 3. We have consistent improvement in accuracy for AlexNet [13], ResNet-18 [9] over ImageNet dataset. The model trained with our proposed two-phase training procedure not only significantly outperforms ReLU but also other non-linear activation function such as PReLU, ELU, and SoftPlus (Table 3).

6.3. PASCAL VOC

We have performed experiments on the SSD model over PASCAL VOC [5] dataset to validate our proposed approach for the object detection task. In this experiment, we follow the same experimental setting and training schedule, as described in [17] for Phase-2 training. For Phase-1 (Cooperative) training, we have used 20% iterations of Phase-2 training with PReLU, ReLU, ELU, and SoftPlus activation functions (having equal weight). The SSD [17] detection model is a feed-forward convolutional network that produces a collection of fixed-size bounding boxes and predicts

Model	Activation Function	Accuracy(%)
ResNet-56 (Baseline)	ReLU	93.5
ResNet-56 (Baseline)	PReLU	94.0
ResNet-56 (Baseline)	SoftPlus	92.0
ResNet-56 (Baseline)	ELU	92.0
ResNet-56 (Baseline-TPT)	ReLU	93.6
ResNet-56 (Baseline-TPT)	PReLU	94.0
ResNet-56 (Baseline-TPT)	SoftPlus	92.0
ResNet-56 (Baseline-TPT)	ELU	92.1
ResNet-56	WNLA	40.0
ResNet-56	Mixture	94.3
ResNet-56	Mix (PReLU)	94.1
ResNet-56	Mix (SoftPlus)	93.1
ResNet-56	Mix (ELU)	93.2
ResNet-56 (Ours)	**Mix (ReLU)**	**94.4**

Table 5. Classification accuracies for ResNet-56 on CIFAR 10. Activation Function: Mixture means model is trained completely using a mixture (ReLU, PReLU, ELU, and SoftPlus) of Activation Functions. Activation Function: Mix (γ) means model is trained using a mixture of Activation Functions in Phase-1 (Cooperative) training and then in Phase-2 model is trained using γ Activation Function. Baseline-TPT means baseline uses Two Phase Training.

#Epochs in Phase-1	Activation Function	Accuracy(%)
10%	Mix (ReLU)	94.03
20%	Mix (ReLU)	**94.40**
30%	Mix (ReLU)	94.40
40%	Mix (ReLU)	94.42

Table 6. The table shows results for changing the number of Epochs in Phase-1 training for ResNet-56 model on CIFAR 10.

classification scores to represent the presence of object class instances in these boxes, followed by a non-maximum suppression which produces the final detections.

As shown in Table 4, our proposed approach is not limited to classification but also works well on object detection task. We have a significant improvement (approx. 1%) in mAP as compare to baseline by using our training procedure.

7. Ablation Studies

As shown in Table 5, if we train a model without any non-linear activation function (WNLA), the performance of the model degrades massively since ResNet-56 is a deep CNN model which is very hard to optimize without any non-linear activation function. There is a performance boost in the case of Mix (SoftPlus) and Mix (ELU) from their respective baselines, but the overall performance scores are significantly lower than Mix (ReLU). The key reason for the significant difference in the performance can be due to the sparse behavior of (ReLU) activation function in Phase-2 training. This sparsity is often desirable in the deep network due to better predictive power and less overfitting/noise.

Although Mixture and Mix (ReLU) have similar performance, Mix (ReLU) should be preferred because of the following reasons:

- Mixture will occupy more feature maps memory as compare to Mix (ReLU) at run time because separate feature maps will be generated for every non-linear activation function. Mix (ReLU) will not increase in feature maps memory at a run time (GPU Memory).
- Mixture will add some delay at the inference time as compare to Mix (ReLU) due to extra calculations performed by multiple activation functions.

Hence, Mix (ReLU) is the most suitable combination while considering all the other possibilities. Our proposed approach uses Two-Phase Training (TPT), where Phase-1 training uses 20% of Phase-2 epochs. Therefore, one may think that performance improvement is due to more training epochs (20%). Hence we also train baselines with the same Two-Phase Training schedule (Baseline-TPT), except in Phase-1, only a single activation function is used. As shown in Table 5, Baseline-TPT has a similar performance as the baseline. From Table 5, we can conclude that performance improvement is not due to more training epochs (20%) in Two-Phase Training (TPT) but because of *cooperative initialization* in Phase-1.

We also conduct an ablation to decide the number of Epochs in Phase-1 training. For Phase-1 (Cooperative) training, we use 10-40% of epochs used in standard (Phase-2) training. As shown in Table 6, 20% of Phase-2 epochs in Phase-1 (Cooperative) training is the most suitable choice because it gives significant performance improvement with only 20% increase in overall training time.

8. Visualizing Last Layer Features on MNIST

In the Figure 2, we are plotting t-SNE [19] plots for LeNet-like network on MNIST [16] dataset to visualize the features learned for various non linear activation functions. The LeNet-like network contains two convolutional layers and one fully-connected layer. The convolutional layers have 5x5 kernel size while the first and second convolutional layer consists of twenty and thirty number of filters respectively.

The t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality reduction technique that is heavily used to visualize the high-level features learned by CNN. The t-SNE is a commonly used technique to visualize feature representations in high-dimensional data into a space of two or three dimensions. From Figure 2, we can visualize the two-dimensional embeddings of the last layer. The features corresponding to Mix (Relu) are more separable than remaining other embeddings. The features corresponding to Mix (ReLU) are well separated and discriminated enough, as shown using corresponding representative points in Figure 2.

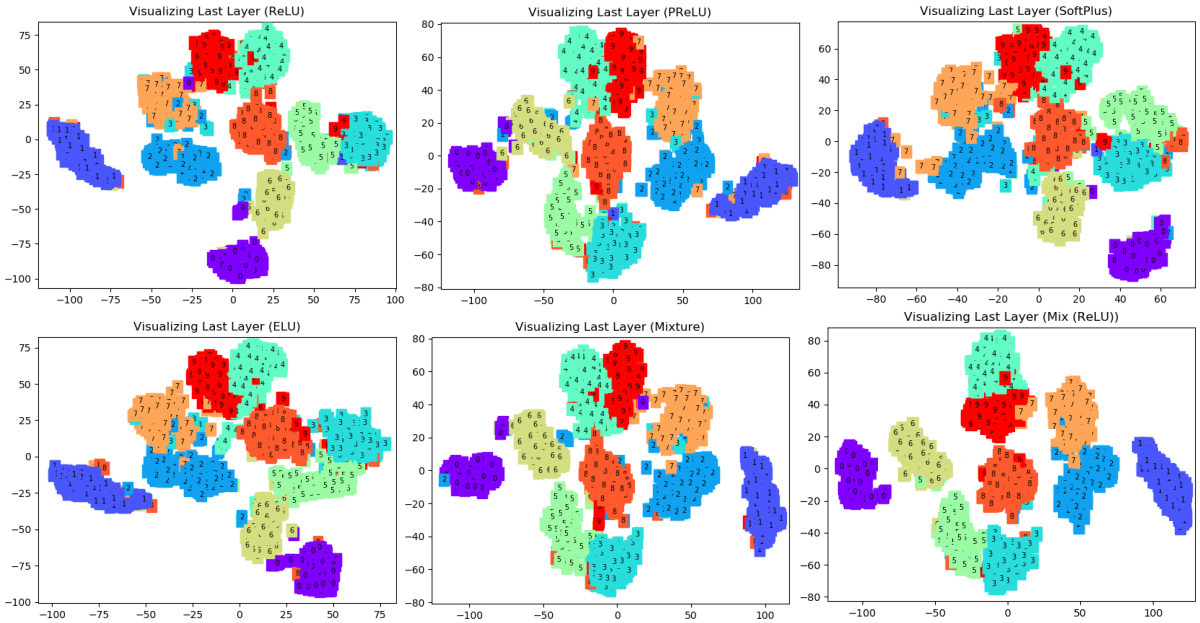

Figure 2. Two-dimensional t-SNE visualization of trained flatten layer features for LeNet-like network on MNIST (Best viewed in color).

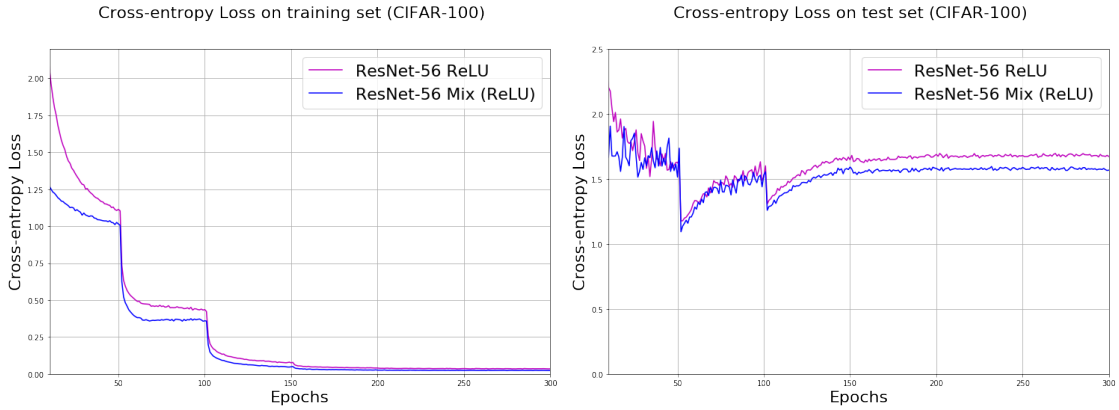

Figure 3. ResNet-56 ReLU and ResNet-56 Mix (ReLU) training and test losses over CIFAR-100 dataset. ResNet-56 Mix (ReLU) exhibits improved optimisation/convergence characteristics and produces significant gain in performance (Best viewed in color).

9. Analysis

This section focused on the analysis of performance gain from two different perspectives. The first perspective focus on investigating the convergence of Mix (ReLU) and ReLU on the ResNet-56 architecture. The convergence speed of Mix (ReLU) is much faster as compared to ReLU, which can be inferred in Figure-3. The second perspective for investigation is based on over-fitting aspects of the models where Mix (ReLU) is more robust compared to ReLU based on the empirical results.

The investigation is performed on the CIFAR-100 dataset using the ResNet-56 model. We used standard data augmentation techniques such as random horizontal flipping and random cropping to size 32×32. The optimization is performed using Stochastic Gradient Descent (SGD) with momentum set to 0.9 and weight decay as 0.0001. The minibatch size of 64 is selected to perform experiments. The initial learning rate is taken as 0.1, which is then decreased by a factor of 5 after every 50 epoch. The models are trained from scratch for around 300 epochs.

9.1. Effect of using Mix (ReLU) on convergence

We analyzed the convergence rate of Mix (ReLU) based model, and we found that the convergence using Mix (ReLU) is slightly better compared to ReLU, which can be inferred from their respective curves in Figure 3. The two graphs of cross-entropy losses vs. the number of epochs for training and test set in Figure-3, shows that the dropping rate of cross-entropy losses is quite higher as compared to the loss corresponding to ReLU on the training set in the experimental results shown in Figure-3.

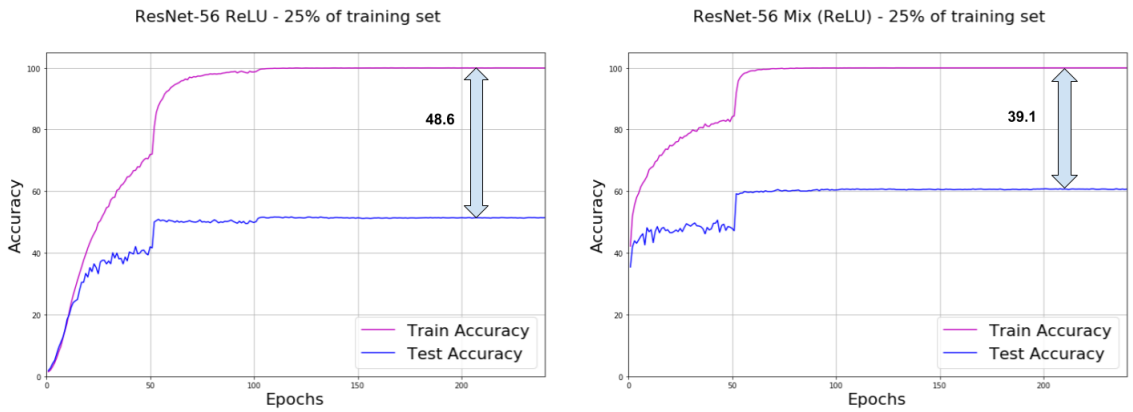

Figure 4. Training curves for ResNet-56 ReLU and ResNet-56 Mix (ReLU) models over CIFAR-100 dataset (Best viewed in color).

Method	Activation Function	Train Accuracy(%)	Test Accuracy(%)
ResNet-56 Baseline (100% of training set)	ReLU	99.4	71.6
ResNet-56 (100% of training set)	**Mix (ReLU)**	99.6	**73.1**
ResNet-56 Baseline (25% of training set)	ReLU	99.9	51.3
ResNet-56 (25% of training set)	**Mix (ReLU)**	99.9	**60.8**

Table 7. The table shows the results for ResNet-56 ReLU and ResNet-56 Mix (ReLU) over CIFAR-100 dataset in different setups.

9.2. Effect of Mix (ReLU) on over fitting

Our method utilize multiple activation functions in the initial few training epochs of the deep network. The gradient from these multiple activation functions gets accumulated and gave a *regularization effect* to the gradient flow in the model corresponding to the complete range of input data (negative and positive). This aggregation of gradients also helps in the regularization of weight parameters while updating, which in effect knock-down the chances of over-fitting issues. In support of our hypothesis, we have performed some experiments in two different scenarios. In the first scenario, experiments are performed out over the complete dataset (100 % of training data), while the second scenario of experiments is performed on only 25 % of the training data.

The first scenario of experiments is performed on the CIFAR-100 dataset using ResNet-56 architecture. The first experiment with only ReLU achieved an accuracy of 71.6% as presented in Table-7 while the second experiment uses Mix (ReLU) and achieve 73.1% accuracy. From Table-7, we can infer that Mix (ReLU) is more robust to overfitting as the difference between test and training accuracy is 26.5, which is lesser than the difference between test and training accuracy of ReLU (27.8). In these experiments, the difference between test and training accuracy is not that much significant for ReLU and Mix (ReLU), hence in the second scenario, we have chosen only 25% samples from the training images of CIFAR-100 dataset to perform various experiments.

The second scenario of experiments is performed to train the ResNet-56 ReLU model using only 25% train samples, and we achieve accuracy of 99.9% and 51.3% corresponding to train and test data respectively. The ResNet-56 Mix (ReLU), on the other hand, achieves 99.9% and 60.8% of training and test accuracy respectively.

From Figure-4, We can conclude that the Mix (ReLU) is less prone to overfitting issues, as the difference between test and training accuracy is 39.1 while this difference is quite higher than that of ReLU, i.e., 48.6.

10. Conclusion

We propose a Cooperative Initialization for training deep networks to improve the performance. We have shown experimentally that a mixture of the non-linear activation function is beneficial for CNN in the initial phase of training, where we start from random initialization. Initially, multiple activation functions regularize the gradient flow corresponding to the positive and negative input of activation functions, thereby improving the update of weight parameters, which is very crucial at the initial stage. Our experimental results show that the proposed approach improves the performances of state-of-the-art networks. Our proposed approach also helps in reducing the overfitting problem and does not increase the number of parameters, inference (test) time in the final model while improving the performance. Therefore, cooperative initialization is a promising approach to improve the feature representation and performance of deep networks.

References

[1] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

[2] C. M. Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[4] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[5] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[6] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

[7] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] G. E. Hinton and R. R. Salakhutdinov. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614, 2009.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 2015.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[18] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models.

[19] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[20] P. Mazumder, P. Singh, and V. Namboodiri. Cpwc: Contextual point wise convolution for object recognition. *arXiv preprint arXiv:1910.09643*, 2019.

[21] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 1943.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] P. Singh, V. S. R. Kadi, and V. P. Namboodiri. Falf convnets: Fatuous auxiliary loss based filter-pruning for efficient deep cnns. *Image and Vision Computing*, page 103857, 2019.

[26] P. Singh, V. S. R. Kadi, N. Verma, and V. P. Namboodiri. Stability based filter pruning for accelerating deep cnns. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1166–1174. IEEE, 2019.

[27] P. Singh, R. Manikandan, N. Matiyali, and V. Namboodiri. Multi-layer pruning framework for compressing single shot multibox detector. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1318–1327. IEEE, 2019.

[28] P. Singh, P. Mazumder, and V. P. Namboodiri. Accuracy booster: Performance boosting using feature map recalibration. *arXiv preprint arXiv:1903.04407*, 2019.

[29] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Leveraging filter correlations for deep model compression. *arXiv preprint arXiv:1811.10559*, 2018.

[30] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Hetconv: Beyond homogeneous convolution kernels for deep cnns. *International Journal of Computer Vision*, pages 1–21, 2019.

[31] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Hetconv: Heterogeneous kernel-based convolutions for deep cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4835–4844, 2019.

[32] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[34] L. Trottier, P. Gigu, B. Chaib-draa, et al. Parametric exponential linear unit for deep convolutional neural networks. In *ICMLA*, 2017.

[35] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[36] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li. Improving deep neural networks using softplus units. In *IJCNN*.